HAAGA-HELIA
University of Applied Sciences

Football Team Tracker System – A Web Application with Agile Development Methodology

Brahim Hadi

Bachelor Thesis
DP in Business Information Technology
2009

Author Brahim Hadi	Group
The title of the thesis Football Team Tracker System - A Web Application with Agile Development Methodology.	Number of pages and appendices 28 + 4
Supervisor Juhani Välimäki	

Often, amateur sport teams try to find an inexpensive way on how they can effectively track and manage their team. FC Tigers football team is one of them. This team has about 30 players working in the same company. The managers of this team needed a practical system to help them keeping track of players available for playing a football game every Thursday evening.

The objective of this thesis consists of two parts: the practical part and the theoretical part. The practical part focuses on building a Football Team Tracker (FTT) System which is a Web Application implemented using Ruby on Rails language and Agile Development Methodology. This application aims mainly to solve the problems with managing a football team. The theoretical part focuses on testing the Feature Driven Development (FDD) Agile methodology against what it has been said about its benefits such as detecting bugs earlier, aims for minimizing documentation, saves you time, iterative approach and so on.

The FTT application provides the basic and the most important features. Due to the lack of time the nice-to-have features and usability are left for the other developers who would like to enhance this system in the future. However, open sourcing this web application and the way this application is designed provide high scalability. So, it can be re-used for any kind of sport which requires a team commitment. One can change the banner's text of this web application and he is ready to go.

As a result for the theoretical part of this project, it has been found that the features of the FDD method tested reflect, nevertheless, what has been said about its benefits. It also fits well within a small Rails project with even two developers.

Key words
Ruby on Rails, Agile Development Methodology, Feature Driven development, VMC model

1 Introduction

FC Tigers football team has about 30 players working at Accenture Ltd. However, depending on their schedule, not all the players are available to play football every week. And sending ping-pong email or calling each player and asking him if he is coming next week to play is costly and time prone. So, the managers of this football team were thinking how they can step over these issues, hence the idea of building the Football Team Tracker (FTT) system came up.

1.1 Objectives of the thesis

A system will be build for the FC Tigers team Managers in order to help them with managing the team. At the same time the Feature Driven Development Agile Methodology is used during the implementation process and tested against the statements that had been said about its benefits.

1.2 Structure of the thesis

This thesis consists of two parts, theoretical and practical. At the beginning of the theoretical part a short briefing will be provided about agile software development methodology in general. Then, some of the most used agile methods and practices will be introduced.

The practical part of the thesis will describe the development process and the implementation of the FTT application. Based on the analysis mentioned in the theoretical part, some of the agile practices will be selected and used in the development process of the FTT application.

After accomplishing the FTT system, the results concerning the agile methods and practices are analyzed and recommendations are given for software developers who would like to use these methods. At the end, a conclusion is given concerning the FTT application and the use of agile methods in small Rails projects.

2 Agile Software Development

Agile software development was introduced for the first time in February 2001 in a manifesto signed by 17 software developers when they were in a ski station in the Wasatach mountains of Utah. (Highsmith, 2001). Since this date the word agile started to become fashion among software architects and developers.

Agile development methodology is well defined in the Agile manifesto as a set of best practices which aim to rapidly adapt to changes along the project processes. At the same time it aims to develop high quality defect-free software and rather focus on coding than on the documentation (Hunt, 2006, 9-12). It is iterative, incremental, documentation minimizing, self-organizing team, doing just what it's supposed to do and enables the knowledge sharing within the team members (Holcombe, 2008, 330).

The most commonly used agile methods are:

2.1 Extreme Programming (XP)

It is the most commonly used agile methodology. Unlike the other agile methods XP is the only agile method which focuses mostly on the programming side. Therefore, almost every member involved in an XP project is programmer. XP is a testing –centric agile method. Even though other agile methods also use the testing component, XP defines how the testing should be done in an XP project. Further more XP is based on communication, simplicity, feedback, and courage (Schuh, 2005.).

2.1.1 Practices

XP includes 12 known practices which can be described as follow:

The Planning Game consists of a close collaboration between the customer and the developers. During the planning game, for each iteration, the customer decides what system feature need to be prioritized and the developers determine the feasibility and the necessary effort needed for building the functionality.

Small releases aim to build a set of features which can be implemented in a period of two weeks. The benefit of the small release is that it can be easily demonstrated to the customer and give a better idea on the project progression.

Test-driven development aims to test before start coding. " Advocates of TDD argue that writing tests before writing code ensures that the programmer knows what code he wants to write, how he wants to use it, and how he will be able to test it." (Schuh, 2005).

XP enquires the commitment of the whole team. This means that the customer, the development team and the management should work in the same workspace in order to facilitate the communication process.

Pair programming is specific for XP. It involves two programmers working at the same work station. They analyze, design and write the code together. New pieces of code are added to the code base as soon as they get ready (Schuh, 2005).Whenever a new piece of code is ready it is integrated to the code-base after being tested.

In XP the code is regularly refactored in order to avoid any code duplicate or to simplify any confusion. So the code is always kept simple and clean to ensure the flexibility and accessibility by other programmers.

The design should be kept as simple as possible to avoid complexity and extra code. This enables the code of the system to work today and in the future because the future might change against the project plan.

In XP a Metaphor is a sort of a user guide of the system created by the customer and the development team. It defines what the system does and how it does it.

In an XP project the code is collectively owned by the whole team. This means that every team member can work on any part of the code at any time in order to complete the task assigned to him.

Coding standard ensures consistency and makes the code easy to be understood by any other developer.

Sustainable pace: sometimes it's called as 40-hours workweek. It focuses on preventing doing over time and working too hard. Instead it encourages the team members to work when they are in a good mood and let them rest when they are tired.

2.1.2 XP process

An XP project consists of a set of iterations. Each iteration takes approximately 1 - 4 weeks. Starting an iteration requires three steps which are: writing stories by the customer and estimating them by the team, planning game and iteration planning game as shown in the figure 1.

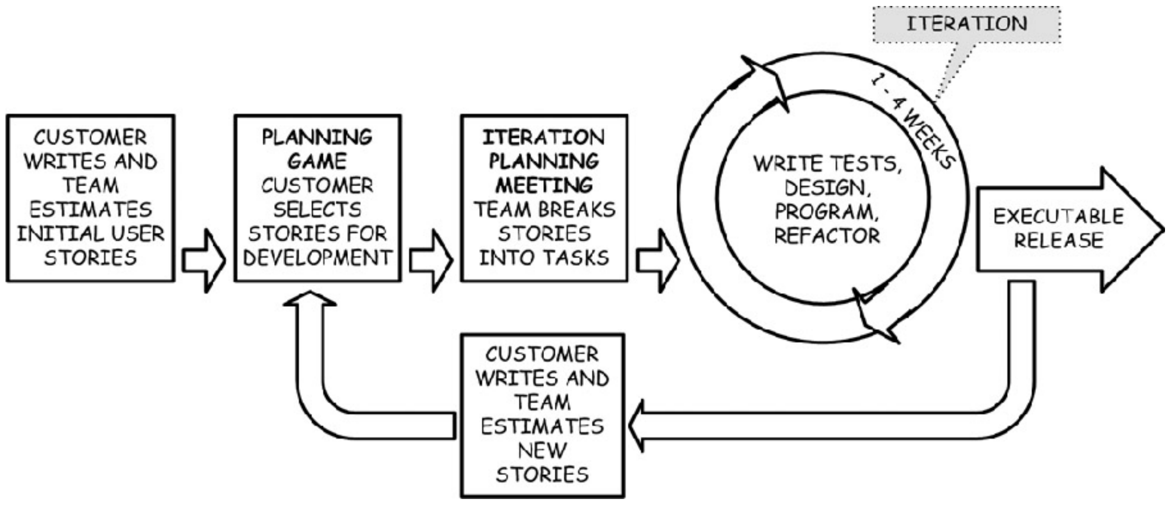

ITERATION
CUSTOMER WRITES AND TEAM ESTIMATES INITIAL USER STORIES
PLANNING GAME CUSTOMER SELECTS STORIES FOR DEVELOPMENT
ITERATION PLANNING MEETING TEAM BREAKS STORIES INTO TASKS
1 - 4 WEEKS
WRITE TESTS, DESIGN, PROGRAM, REFACTOR
EXECUTABLE RELEASE
CUSTOMER WRITES AND TEAM ESTIMATES NEW STORIES

Figure 1: XP process (Schuh, 2005)

During an iteration the developers implements a prioritized feature list. The implementation includes writing test, design, program and code refactoring. When the iteration is complete the team delivers a fully tested, functional release and worth to be in production (Schuh, 2005).

2.2 Scrum

Unlike other agile software development methods Scrum is a management process which can be applied in different activities where the commitment of a team work is required.
Within the rules of this method a project is divided into features which are prioritized and categorized into sprint backlogs.

A daily scrum meeting of about 15 minutes is held. During this meeting every team member should mention what he has achieved so far, any problems that had emerged and what he is going to do next. The purpose of these short meetings is to monitor the progress and react to any potential problems or changes as fast as possible.

After each sprint a review meeting is kept in order to focus on the quality of the features implemented. Among the actors involved during the process of this method are: the product owner, the scrum master, and the team members (Holcombe, 2008).

2.3 Dynamic System Development Method

The Dynamic System Development Method is an approach which focuses more on how to build a software system than on how to manage a software project (Koch, 2005.).
It consists of involving the users in the project life cycle so that decisions can be made as accurately as possible. The incremental iterations are based on prototyping (Holcombe, 2008.).

There are nine principals which define this method:

1- Active user involvement is imperative.

2- DSDM teams must be empowered to make decisions.

3- The focus is on frequent delivery of products.

4- Fitness for business purpose is the essential criterion for acceptance of deliverables.

5- Iterative and incremental development is necessary to converge on an accurate business solution.

6- All changes during development are reversible.

7- Requirements are baselined at a high level.

8- Testing is integrated throughout the life cycle.

9- A collaborative and cooperative approach between all stakeholders is essential (Koch, 2005.).

The process of this method includes four components: Feasibility and business study, functional model, design and build and Implementation as shown in figure 2 (Koch, 2005.).

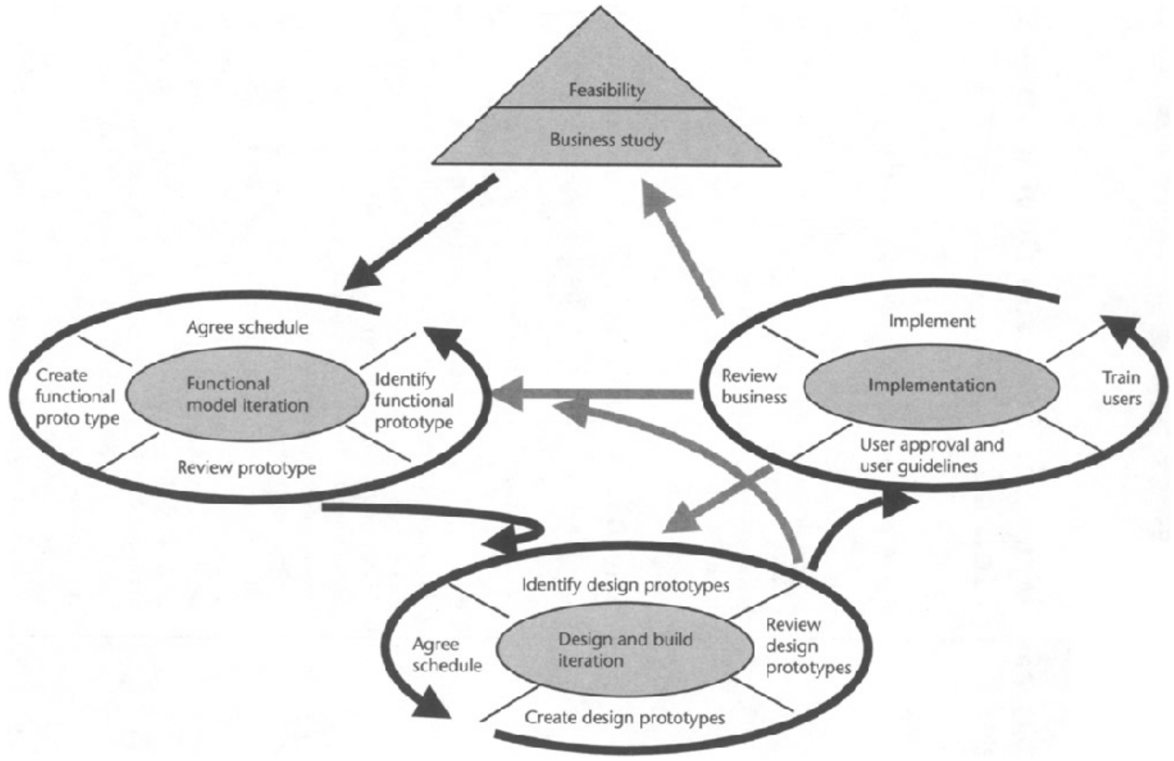

Feasibility
Business study
Agree schedule
Create functional proto type
Functional model iteration
Identify functional prototype
Review prototype
Implement
Review business
Implementation
Train users
User approval and user guidelines
Identify design prototypes
Agree schedule
Design and build iteration
Review design prototypes
Create design prototypes

Figure 2: DSDM process diagram (Koch, 2005.).

2.4 Feature-Driven Development (FDD)

Before starting to enter into details of this method let's try to define what is a feature?

A feature is the concept of a piece of functionality that:

- Represents a customer experience within the system
- Once completed, will be accessible to and usable by the customer (notwithstanding the completion of any other features that it might be dependent upon)
- Can be estimated by the project team
- Does not take more than a few weeks to complete (Schuh, 2005, 168).

Feature-driven development (sometimes called feature-driven design) sees a software development project as a set of features which are implemented incrementally one by one. Compared to the other agile methods, FDD starts by creating a domain object model which is used as the main document within the communication process. The domain object model is created by the developers in collaboration with the domain expert. This document is also used to identify the features list. On the other hand, it provides an effective monitoring mechanism for reporting the project progress against the project plan. (Holcombe, 2008.). According to Schuh (2005, 26), an FDD project is highly iterative and involves the customer in every iteration, through out the whole life cycle of the project.

2.4.1 Practices

FDD provides eight best practices which are:

Domain object modeling: As discussed earlier, it is the starting point of an FDD project. A domain object model is a general roadmap of the whole system. “It consists of a handful of high-level diagrams that depict the relationship between classes and sequence diagrams that demonstrate behavior” (Schuh, 2005, 26.).

Develop by feature: In every agile project the software to be developed is seen as a set of features which are built one by one.

Class ownership: Each class of the system is owned by a specific programmer while in XP the code is collectively owned by the whole team.

Feature teams: Since features usually involve many classes, feature teams are necessary to design and implement in FDD. Feature teams are usually small teams formed dynamically.

Inspections: It focuses on detecting defects without trying to intimidate the programmer. This includes improved knowledge sharing among the developers.

Regular build schedule: The entire system is built regularly in order to make it available for testing or for the client demos. Depending on the need, the system might be build hourly, daily or weekly.

Configuration management: Using a versioning tool the code is stored and versioned depending on the need of the team.

Reporting/visibility of results: The results and the updates of the project status are regularly reported in order to keep the stakeholders aware of the current situation. This is very important for the stakeholders so that they can react accordingly. (Schuh, 2005, 26-27).

2.4.2 Process

The process of this agile method consists of five components as mentioned in the following figure:

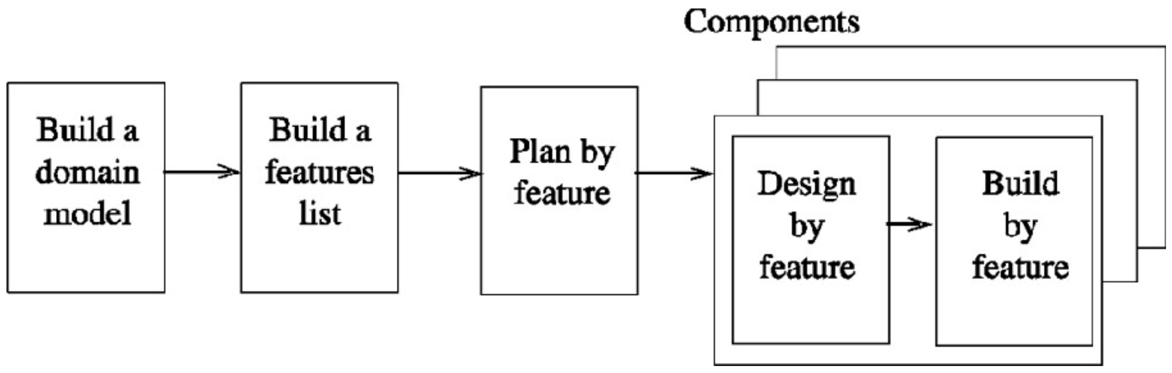


Figure 3: Feature-driven development process (Holcombe, 2008.)

Since FDD method is used in the practical part of this thesis the details of its process is given later in the development process chapter.

2.5 Comparing the agile methods and choosing the method for this project

The following table shows the most common characteristics of the above mentioned agile methods and the difference between them.

Table 1: Summary of the Features of the Agile Methodologies (adapted from Mike Holcombe, 2008)

Feature	DSDM	FDD	XP	SCRUM
Feature centric	+	+	+	+
Clear business focus	+	+	+	+
Strong quality/testing focus	?	–	+	?
Handles changing requirements	+	+	?	+
Human-centered philosophy	?	?	?	?
Support for maintenance	?	–	–	–
User/customer-centered approach	+	+	+	?
Encourages good communications	+	?	+	+
Minimum bureaucracy	?	?	?	–
Support for planning	+	+	+	+

It is common to all agile methods that they tend to complete the software project by feature as they tend to handle unexpected changes instead of avoiding them. However, they are different in some aspects as it's shown in the previous table.

After reviewing and examining the agile methods it has been concluded so that the FDD method is the most suitable method for this project due to its smallness, simplicity and the small amount of developers committed. Besides, it has been decided so that it won't be any extensive testing for the software. Hence, the FDD method has been chosen as a candidate agile method to be tested in this project with some tailoring and adjustments.

3 Case: Football Team Tracker Application

3.1 Introduction

This practical part of this thesis will be devoted for the development process of the Football Team Tracker Application (FTT). This application is build using Ruby on Rails (RoR) language. In the implementation phase the book titled Agile Development with Rails (Thomas & Heinemeier Hansson, 2006, 719 pages) is used as a reference and a guide. As this language is relatively new for the developers of the FTT application some component of RoR and the implementation of most important features will be explained.
During the development process the FDD agile method is chosen as it seems to be the most suitable agile method for a small project.
The steering group of this project is quite small and consists of only: Sponsor, project manager, two developers and the advisor.
The main goal of this thesis is to build the FTT system and at the same time the FDD agile method will be used and tested in order to find out whether it provides the benefits claimed and fits well within a small Rails project. The best practices of the other agile methods will be used only if necessary.
At the end of this thesis a discussion about this method will be provided.

3.2 Background

Football Team Tracker (FTT) System is a free open-sourced web application, coded using Ruby on Rails (RoR), which is supposed to act as a helping hand in tracking and managing the football players of FC Tigers Team.

This system will help managers of this team to find available players (10-15 players) to play a football game every Thursday at 07:00 PM. So, no more time wasted calling or emailing people.
FTT system should offer the possibility of keeping track of attendance, automatic confirmation emails before the game, cost tracking and billing function (to pay for sport venue rental, equipments, tournament inscriptions and other such payments).

The application is made easy to use. Only the players and the managers of the FC Tigers team can log into the system and use its functionalities.

There are two views depending on the type of users. The admin view has most important functionalities that a system administrator would need, such as Delete, Add and Update records. However, the player view is limited and allows only what a player needs to do.

3.3 Development process

During the development process mainly the FDD method is used. However some best practices of the other methods are used whenever necessary with some tailoring.

3.3.1 Developing an overall model

During the kick-off meeting of the FTT project the domain experts who consisted of the managers of the FC Tigers team explained the issues which were facing with the team management and gave an over-all idea on what the FTT system should do. The FTT project manager collected the maximum of data during that meeting. One week after, a prototype version of the FTT system was released using Microsoft Powerpoint slides. This prototype was modified a couple of times in order to fit the customer's requirements.

3.3.2 Identifying the feature list

Based on the prototype, a list of 20 features was identified. This list contained the most client valued functions of the FTT system. This features list was reviewed by the sponsors and users of the FTT system in order to make sure the list is complete and valid. This features list is attached further in the appendices section.

3.3.3 Plan by feature

At this stage the most complicated features were divided into small sub-features so that the implementation can be easier for the developers. The features list was prioritized so that the must-to-have features are of priority one, should-be-there features of priority 2 and nice-to-be-there with priority 3. The target was so that at least the first two feature categories are implemented. However, the third one can be implemented only if there will be time for it.
Based on this list the skills and resources were identified and the schedule was set to the Project.

3.3.4 Design and build by feature

Every Sunday, a backlog was set and published with prioritized sub-features. These sub-features were assigned to the developers based on the required skill. The developers put the hours needed for the tasks assigned to them and start working on for a maximum period of 7 days. The testing of new functionalities was done simultaneously in the local environment in order to detect any potential bugs as earlier as possible.

Whenever a feature or a sub feature is implemented it is promptly uploaded to the Google code's subversion repository at: http://code.google.com/p/sport-team-tracker/ so that it can be immediately tested by the stakeholders and come up with feedback if any.
On the top of that the customer could ask for any change request at any stage of this project. The bugs were fixed during the following backlog. However, the change requests were first discussed with the customer and then validated as soon as possible so that they could be included in the feature list and start working on them.

3.4 Development environment

In this project every component of the development environment is open sourced.

3.4.1 Ruby on Rails

Ruby on Rails (sometimes shortened as Rails or RoR) is a web application development framework made using ruby programming language.
It has been said that Rails itself is agile because it favors interaction between individual across processes, working software, customer collaboration through contract negotiation, use of comprehensive and trivial documentation, and responding to changes along a project plan. Rails doesn't require heavy tools, or a complex configurations. Instead there are small teams of developers, their favorite IDE/editor and pieces of Ruby code (Thomas & Heinemeier Hansson, 2006, 3.).

3.4.2 Ruby

The FTT application is written using Ruby language. So what is this language?
It is a dynamic object programming language invented for the first time in Japan by Yukihiro Matsumoto during the mid-1990s. Ruby language is classified as a scripting language. This

means that the Ruby code is not compiled in order to be run. However, the code is converted by the interpreter into a format that the computer can execute at run time.
It is an open source project and continuously growing in popularity among developers community (Fisher, 2008, 15-18).

3.4.3 Aptana Studio

In this project Aptana Studio has been used as an Integrated Development Environment (IDE). It is created by Aptana company, based on Eclipse project.
The features of Ruby on Rails are packaged as plug-ins, just like in Eclipse. It also supports CSS, html and java script. In another word, if someone is familiar with Eclipse then he shouldn't have any problem to use Aptana Studio.

3.4.4 TortoiseSVN

As a version control, TortoiseSVN has been used in this project. It can be used in any environment, easy to use and free of charge. It enables storing the source code and gives the possibility to undo mistakes and daily monitor the changes. So, it has been used as a tool to monitor the project status as well.

The FTT application source code is freely available in Google code's subversion repository at: http://code.google.com/p/sport-team-tracker/ .
Downloading the code requires a subversion tool which can be found at:
http://tortoisesvn.net/downloads/

3.5 Implementation

In this part, a brief description will be provided on how the most important components of RoR and how some FTT features have been implemented.

3.5.1 Model view controller (MVC)

The VMC model is an architecture introduced for the first time in 1979 by Trygve Reenskaug. In this model an application is divided into three separate components: models, views and controllers (Thomas & Heinemeier Hansson, 2006, 11.).

The Model represents the database part of the system and handles the business rules. The view's task is to display the user interface to the end user mainly based on the model component. So, the view fetches indirectly the data requested by the user from the model component and renders it back to the user. The controller component plays the role of a connector, a communication facilitator between the two other components. The controller first gets events from a client, communicates with the model component and gets the necessary data from it. Then it invokes the appropriate view which formats the collected data in order to be rendered to the client, as shown in figure 4.

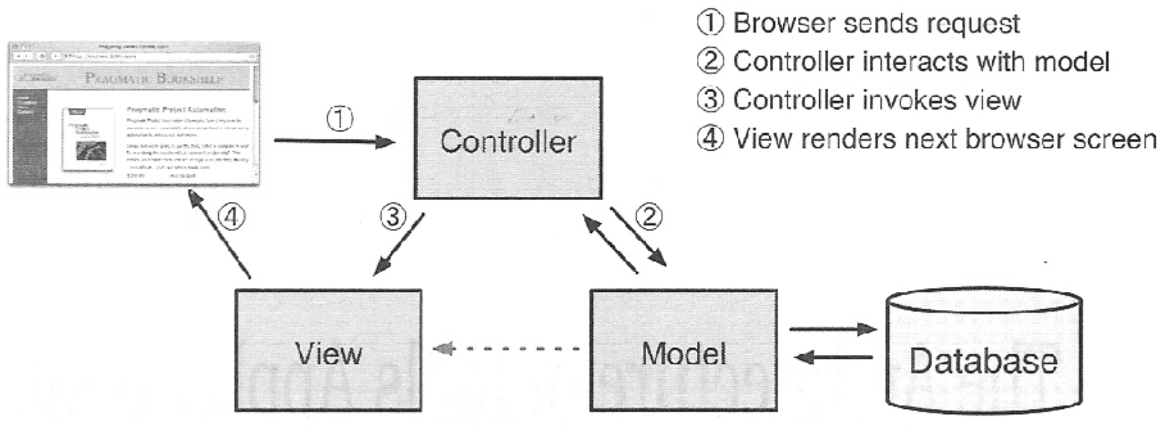

① Browser sends request
② Controller interacts with model
③ Controller invokes view
④ View renders next browser screen
Controller
View
Model
Database
①
②
③
④

Figure 4: The Model-View-Controller Architecture (Thomas & Heinemeier Hansson, 2006, 12.)

The use of the MVC model makes the code of an application easier to write and maintain. Since the user interface is isolated from the business logic a programmer can modify the user interface component without affecting the business logic and vice versa.

Ruby on Rails works according to the VMC model. The three components are well isolated from each other. However, these three components are combined together as the program executes. One of the most advantages of RoR is that the combining mechanism is done automatically and does not require any separate configuration (Thomas & Heinemeier Hansson, 2006, 13.).

3.5.2 Admin and player views

The FTT system consists of two main views: the admin view and the player view. The dedicated database contains a table called "users" which has a column named "role". The value of this role can be ”0” for admin and ”1” for a player. Depending on the user trying to log in, the system checks whether the user is an admin or a player.

The admin user has full control of the FTT system. However, a player is restricted to perform certain tasks such as editing, creating or deleting items. This is done by a simple checking of the current session object using an if-statement. The following piece of code shows how the views are checked when they are requested by an active session:

```
<%if session[:role] == 0 %>
    .
    .
    .
    <td><%= link_to 'Show', :action => 'show', :id => event %></td>
    <td><%= link_to 'Edit', :action => 'edit', :id => event %></td>
    <td><%= link_to 'Delete', { :action => 'destroy', :id => event },
    :confirm => 'Are you sure?', :method => :post %></td>
    <td><%= link_to 'Enroll', { :action => 'enroll', :id => event },
    :method => :post %></td>
    .
    .
    .

<%else%>
    .
    .
    .
    <td><%= link_to 'Enroll', { :action => 'enroll', :id => event },
    :method => :post %></td>
    .
    .
    .

<% end %>
```

So, if the role session object equals to ”0” then the user is an admin and he is given full rights to the whole system. Otherwise the user is a player and then he is prevented from critical tasks such as delete, update and add.

3.5.3 Lay out page and CSS

The layout is part of the views written in pure XHTML. It’s used by all the other pages as a template. It holds the left side bar and the banner. The content is filled by the appropriate view depending on the page requested by the user.

The mechanism of how the layout works is pretty simple. A declaration is put in the top of each controller is enough to call the layout to hold the view related to that particular controller.

CSS is used to have control on the layout page. The CSS file resides in the public folder and it is called only once in the layout by using a link.

3.5.4 Database migration

First Ruby on Rails developers had some difficulties with updating the database using complex SQL queries throughout multiple version of the application being developed. The database migration came as a solution for this problem. It enables you to create the database schema in a Ruby class (extended from ActiveRecord class) using simple ruby code instead of using complex SQL queries.

In this project all the data models have been created using the migration. In order to explain the migration the table users from FTT application is used.

The migration process consists of two simple steps:

1- Creating the migrate file

This can be done in two ways:

a) The migration file (class) can be created while the model is created (unless -- skip-migration is specified) using the following command:

```
ftt> ruby script/generate model user
```

As a result the following files are generated:

```
exists  app/models/
exists  test/unit/
exists  test/fixtures/
create  app/models/player.rb
create  test/unit/player_test.rb
create  test/fixtures/players.yml
exists  db/migrate
create  db/migrate/002_create_users.rb
```

So the model user is created and at the same time the 002_create_users.rb migration class is created as well. This class contains two methods, one for dropping the table if it exists already and the other one is for creating the table itself as it's shown below:

```
class CreateUsers < ActiveRecord::Migration
  def self.up
    create_table :users  do |t|
    end
  end

  def self.down
    drop_table :users
  end
end
```

Columns can be added at any time. For example a user_name column with a string data type is added by running the following command:

```
ftt>ruby script/generate migration add_user_name_column
```

The previous command generates a file like ###_add_user_name_column.rb in the /db/migrate folder. This file contains the up and down class methods which need to be filled manually with add_column and remove_column methods respectively as shown bellow:

```
class AddUserNameColumn < ActiveRecord::Migration
  def self.up
     add_column :users,   :user_name, :string
    end
  end

  def self.down
    remove_column :users,   :user_name
  end
End
```

b) The migration file (class) can be created separately on its own using the following command:

```
ftt> ruby script/generate  migration user
```

As a result the following files will be generated:

```
exists  db/migrate
create  db/migrate/002_users.rb
```

2- Running the migrations

Migrations can be run using the db:migrate Rake task. For example to run the the migration of the FTT we run: ftt>rake db:migrate. Running the migration will create all the necessary tables with all the necessary columns and data types as long as the necessary migration files (classes) are created before hand.

This was a very handy procedure in the FTT software especially when at any time a developer created a new table or modified the data model. In order to synchronize the data model, all what the other developer needed to do was to upload the latest code (using tortoiseSvn version control) and run the migration using rake db:migrate command. After that he was ready to continue coding his tasks assigned to him during a particular sprint.

3.5.5 Action mailer

Action mailer is a component of Rails which allows a rails application sending out and receiving emails. The implementation is simple and consists of three parts:

1- Creating a mailer:

The action mailer can be created using the following script:
./script/generate mailer user_mailer

The script generates a bunch of files spread over the folders of the application. The most important file is the UserMailer class placed in the Models folder.

This class holds two methods: the register method which is responsible for setting up the environment for sending an email to a newly created account for a new player and the method invite is responsible for setting up the environment for sending a broadcast email to all the players when a new event is created. The following code shows the UserMailer class and its methods:

```
class UserMailer < ActionMailer::Base

  def register(user)
    @subject    = 'sport team tracker welcomes you'
    @body ["user"] = user
    @recipients = user.email
    @from       = 'sttracker@fakemail.com'
    @sent_on    = Time.now
  end

  def invite(user)
    @subject    = 'come drible'
    @body ["user"] = user
    @recipients = user.email
    @from       = 'sttracker@fakemail.com'
    @sent_on    = Time.now
  end

end
```

2- Email configuring

The e mail configuration is done in the development.rb in the config/environments directory. In this file we have to specify how the email should be delivered and where to find the SMTP server to handle the outgoing email (Thomas & Heinemeier Hansson, 2006, 567.).

```
config.action_mailer.delivery_method = :smtp

config.action_mailer.server_settings = {
  :tls => true,
  :address => "smtp.gmail.com",
  :port => 587,
  :domain => "sttracker.com",
  :authentication => :plain,
  :user_name => player@gmail.com",

  :password => "diffPw"
  }
```

3- Email templates

The email templates are created automatically when the action mailer is created. These templates are part of the View component. The number of templates is the same as the number of the methods existing in the UserMailer class. These templates can be accessed and modified accordingly depending on the context.

3.6 Project success

A functional Football Team Tracker application has been implemented successfully with basic features. The customer already tested it locally and planned to put it into production environment starting from next season. So, the target of this project has been reached successfully.

The implementation phase did not go as planned because Ruby language was more challenging to learn than expected. The developers were learning by doing, since learning Ruby was one of the preoccupations of the team. Asking the Ruby community was a big help in learning and stepping over some of the barriers faced. As a result the team benefited a lot of knowledge sharing and gained a good base of this new language to them.

Managing the time was a bit problematic due to some personal issues. As a fixture an over time work was required time to time in order to catch the time lost.

The face to face communication has helped the team a lot. It saved them time and avoided misunderstandings between team members.

4 Results and Discussion

As a concrete result of this project a working Football Team Tracker application is implemented and ready to use. During the development process there were some minor problems with managing the time due to some unexpected personal life issues such as sickness. As a lesson learned in this project it has been found out that the project plan could be a bit looser. In another word, when planning the timetable for a project one should not plan it to be too strict time wise but instead plan some time for facing possible problems and solving them. This issue was considered in the project plan but no time was assigned to it.

After applying the Test Driven Development Agile methodology in the development phase of this project many ideas and questions have been raised up. Team members of this project realized that building by feature is a funny way to work. Developers were well committed in the project and strongly felt like they have a regular duty to achieve every week.

Due to the small amount of people involved in this small project it was not possible to measure the communication effectiveness of FDD methods. Mainly emails and wiki page was used as a communication mean and no misunderstanding was notified within the team members of the project and the stake holders. If the size of the project was big, the planning game or scrum meeting would be the best in this case as stated in the chapter 2.1.1 and 1.2.

During this project only two change requests (CR) have been received from the customer. With only two CRs it was not possible to state an objective opinion on how the FDD handles change requests.

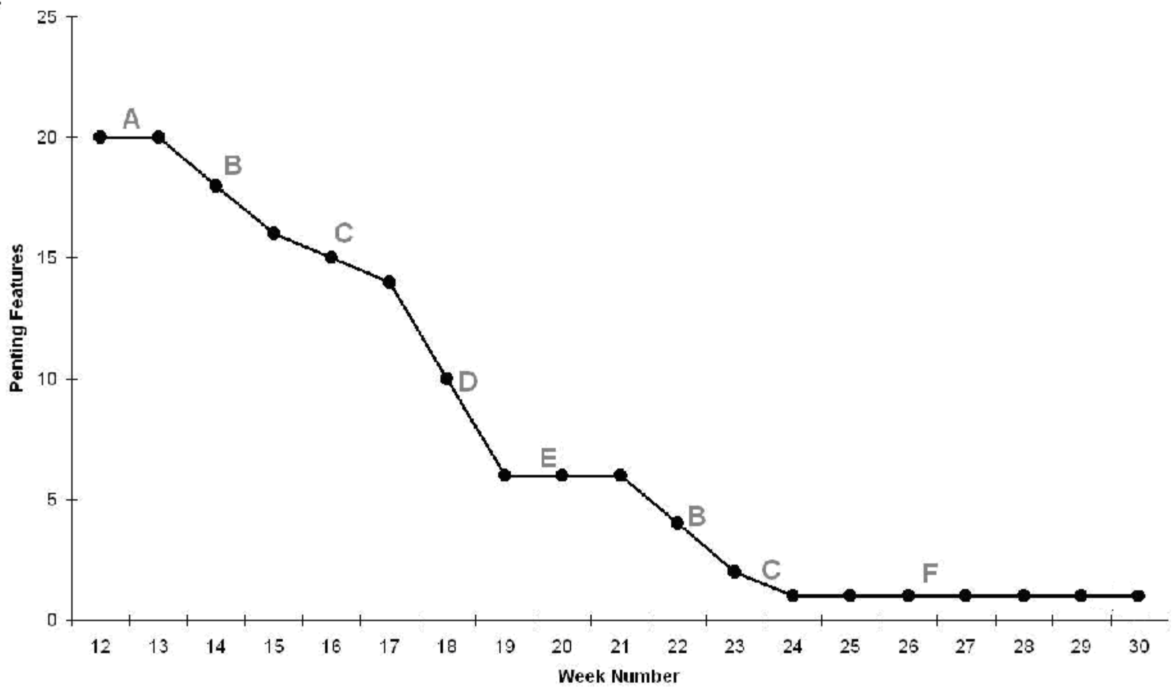
Team members of this project did not feel the need of high level documentation in this small project. They rather used face to face communication and emails. However, using Microsoft Powerpoint, a prototype document has been produced instead of producing an upfront model. At the early stage this prototype document was likely to be incomplete, but still it helped the developers focus on how actions should be sequenced. So, no 500-pages specification was used like in traditional projects.

The project manager did not feel the need of Test Driven Development or an intensive functional testing. However, after each build the new functionalities were tested locally first by the developers and then by the customer after uploading the code of new features to the shared repository. This opinion match perfectly with what it has been stated the in table 1 about testing.

Team members of this project realized that the approach of building by feature gives an accurate way to monitor the progress of the project. Monitoring the FTT project started since the features had been identified.

The FTT project has been broken down into 20 features. Each feature included a set of sub features. A backlog has been set on weekly basis and each backlog included about two features depending on how difficult the feature was to implement and on the availability of the developers during a particular week.

As said earlier, only two CRs have been received during the development phase. The CRs were immediately discussed with the customer, agreed on and then included to the features list accordingly. Those CRs did not cause any slow down of the progress of the project since those CRs were about minor sub features. However, personal issues like illness did cause a delay of the project during the week 15-16 and 23. As a recovery from this delay an overtime work was needed time to time. The graph 1 summarizes the progress of the FTT project.



Graph 1: FTT project progress over time.

- A: Warming up and setting the development environment
- B: Normal rhythm of the project
- C: Illness

- D: Overtime work
- E: Serious sickness of one member of the family
- F: Freezing the development part and switching to the theoretical part of the thesis.

As shown in the previous graph, after each build the project progress can be precisely monitored based on how many features have been implemented and how many features are still pending. If any unexpected risk which might slow down the project progress occurs, it can be easily detected and analyzed. Thus, the stakeholders could continuously be informed about the status of the project progress.

4.1 Recommendation for future research

Integrating the FTT software with a banking system is left for further implementation. As the FTT is an open source one could implement an online payment system within it. This could be very useful, especially if the team committed in a particular sport game is big. There are ready solutions available for integrating a payment gateway in a website such as Paypal (www.paypal.com) internationally or Suomen Verkkomaksut (www.verkkomaksut.fi) in Finland so it is more a question of cost.

The FTT application could also be customized depending on the need of the team. The FTT system itself is easy to customize via CSS to suit the needs of a particular sport team.

Concerning agile methodologies, other methods could be tested as well so that an objective comparison could be concluded. The FDD methods could be tested in more projects because only one project is not a representative sample to state a complete and an ccurate opinion about it. One could also test all these agile methodologies mentioned in the chapter 2 and combine their best practices into one agile method with some tailoring and customization.

5 Conclusion

The main objective of this thesis was to build the FTT web application for the FCT football team. The objective has been reached with minor problems as in any development project. The acceptance testing has been executed locally and the customer was satisfied with the solution. The customer decided to put it into production environment and start using in it starting from next season.

At the same time the Feature Driven Development agile method was tested. However, it is quite difficult to be objective and make an accurate statement about a particular feature of the agile method because projects are unique. But still it was found that the features of the FDD method tested reflect, nevertheless, what it has been said about its benefits. Also, it has been found out that the FDD method suits well for a small development project as it has been claimed in the chapter 1.4.

Due to the small size of this project not all the practices have been tested such as the communication effectiveness between developers and stakeholder.

On the other hand, it has clearly been noticed that no heavy documentation was needed in the development process but just trivial and comprehensive one like a prototype document, database model and the requirements' list. This result reflects exactly what agile methods look at the use of heavy documentation. As mentioned in the chapter 1.4.3, Rails itself is agile and use only trivial and comprehensive documentation.

Building by feature has worked very well in this project. Every week a backlog was created with prioritized features. In this research it has been concluded that building by feature gives a concrete and a measurable way to monitor the project progress.

6 Key terms and acronyms

Aptana Studio - A complete web development environment which allows you to develop, deploy and manage your application in one place.

CR - Change request

FDD - Feature Driven Development Agile method

FTT - Football Team Tracker

Gem - A package format of the Ruby programming language library.

MVC - Model View Controller

Open Source – Software whose source code is freely available and can be modified and redistributed freely.

Rake - is a software build tool written in Ruby programming language

Ruby on Rails (RoR) - is an open source web application framework for the Ruby programming language.

XP - Extreme Programming

7 Bibliography

Fisher, T. 2008. Ruby on Rails Bible. John Wiley & Sons.

Highsmith, J. 2001. History: The Agile Manifesto. http://agilemanifesto.org/history.html. Quoted 23.5.2009.

Holcombe, M. 2008. Running an Agile Software Development Project. John Wiley & Sons.

Hunt , J. 2006. Agile Software Construction. Springer. USA

Koch, Alan S. 2005. Agile Software Development: Evaluating the Methods for Your Organization. Artech House.

Schuh, P. 2005. Integrating Agile Development in the Real World. Cengage Charles River Media.

Thomas, D., Heinemeier Hansson, D. 2006 Agile Web Development with Rails. 2nd Edition. The Pragmatic Bookshelf. USA.

8 Appendices

8.1 Requirements' list

#01: The admin view should give full access to the FTT system. This includes sensitive operation such as create, modify and delete.

#02: All users shall log into the system via a login page

#03: The admin user should be able to create a new event

#04: The admin user should be able to delete an event

#05: The admin user should be able to edit an modify an event

#06: The admin should be able to enroll to an event

#07: An automatic broadcast email should be sent to all players when a new event is created, modified or cancelled.

#08: The admin user should be able to create news

#09: The admin user should be able to modify news

#10: The admin user should be able to delete news

#11: The admin user should be able to create a new player

#12: The admin user should be able to edit and modify a player's data

#13: The admin user should be able to delete a player

#14: An automatic email should be sent to the newly created players informing him about his credentials and the link of the FTT application

#15: All users (admin and players) should be able to view their previous participations in which they should be able to see to which event they have participated, how much money they should have paid so far.

#16: All users should be able to quit the FTT system using a logout link

#17: The player view should prevent non admin users (players) to perform sensitive operation such as create, modify and delete.

#18: A logged player should be able to display events and enroll to suitable one (s).

#19: A logged player should be able to display news, list of players and their contact information.

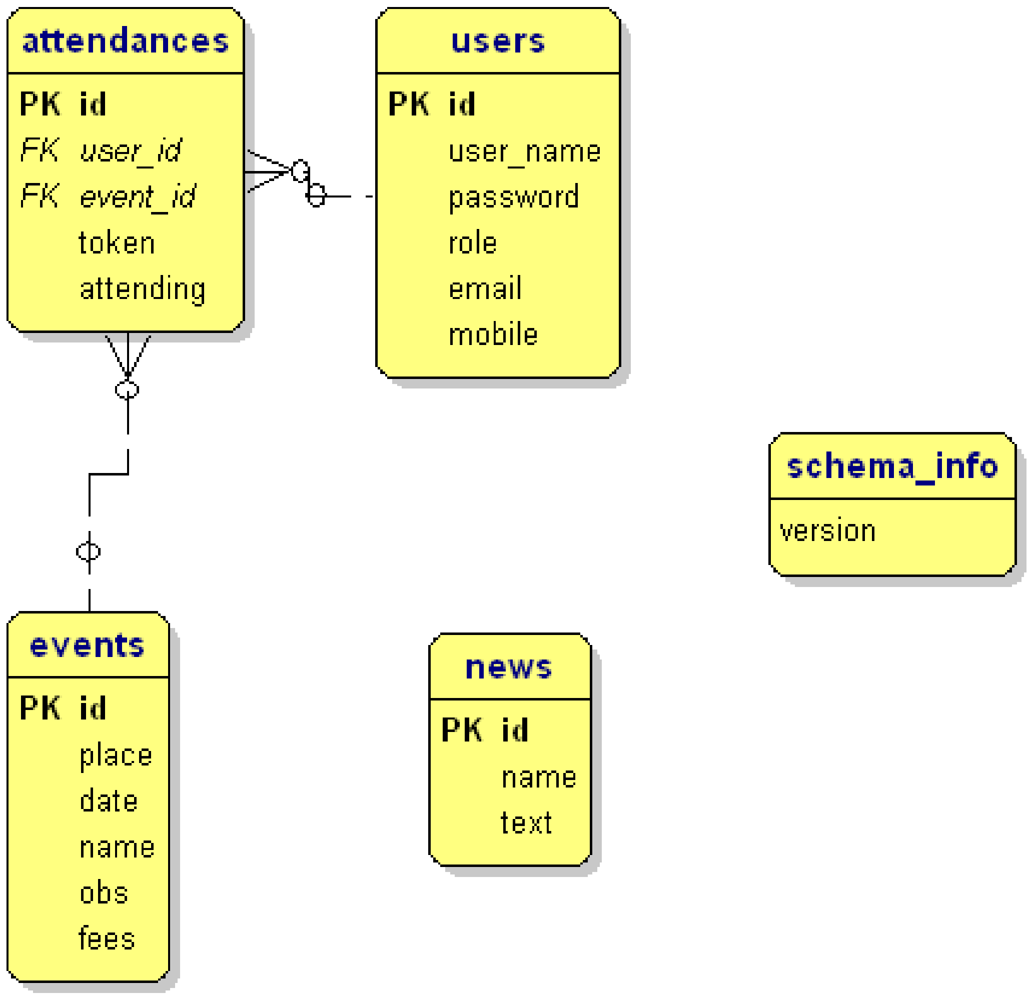
#20: Every 16th of each month, a bill should be sent to each player via an automatic email.

8.2 Change requests

CR #01: Events view should include the fees of each event

CR #02: The logged user should be able to see the total fees that should be paid so far.

8.3 FTT database model



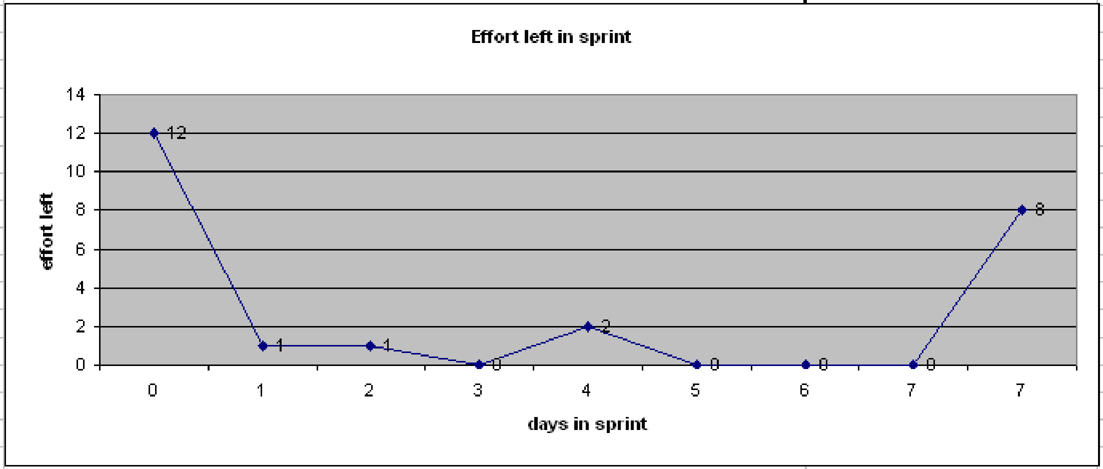
8.4 Ruby on Rails developing environment guide for windows

This guide can be downloaded at: http://sport-team-tracker.googlecode.com/files/RoR_Environment_guide.doc

8.5 Backlog template

Sprint 3. FTT System 03.05 - 10.05

Feature ID	Feature/task	Developer	days in sprint / effort left									Status
			0	1	2	3	4	5	6	7	7	
			12	1	1	0	2	0	0	0	8	
6	**MyBill**											
	The bill as a form should be displayed after hitting the MyBill link. This form should contain at least: To account Number, BeneficiaryName, Amount, Due date and Message as Football fees	Developer 1	3	0	0	0	0	0	0	0	3	Not Started
5	**MyParticipation**											
	Only events that have passed should be displayed. (Maybe MyParticipation should be renamed to MyPreviousParticipations.)	Developer 1	3	0	1	0	0	0	0	0	2	In progress
3	**Events**											
	Only events that will happend in the future should be displayed. (Maybe Events should be renamed to UpcomingEvents.)	Developer 2	3	0	0	0	0		0	0	3	Not Started
3	**Events**											
	Set the number of enrolled players to a particular event to be max=15	Developer 2	3	1	0	0	2		0	0	0	Completed

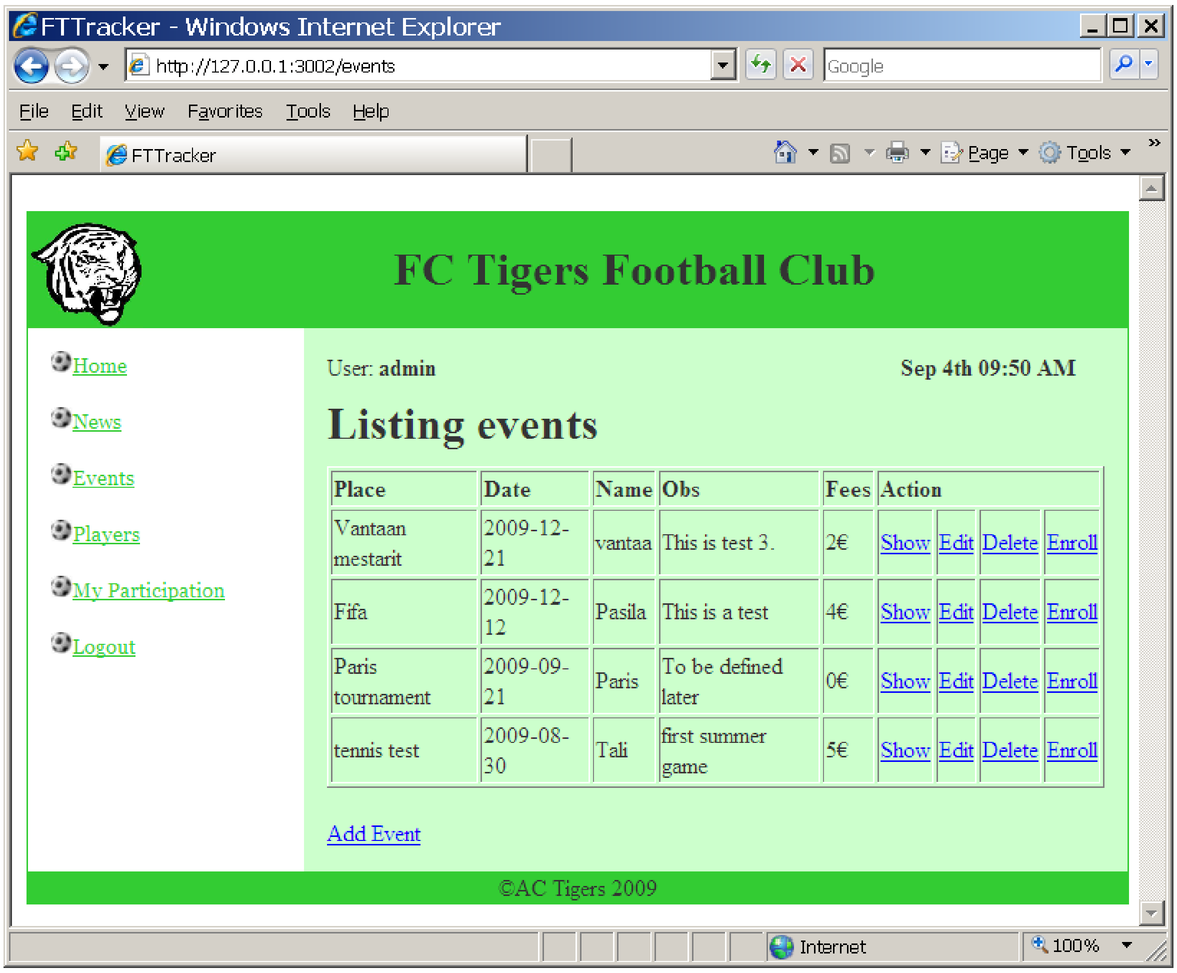


Views	
Id	**Name**
1	Home
2	News
3	Events
4	Players
5	MyParticipation
6	MyBill
7	SendRequest

Backlog state taken after day 7 (from Monday to Sunday)

N.B. Please update the backlog whenever you work on these features.

8.6 Samples from the FTT system



FTTracker - Windows Internet Explorer

http://127.0.0.1:3002/events

FC Tigers Football Club

Home
News
Events
Players
My Participation
Logout

User: **admin** **Sep 4th 09:50 AM**

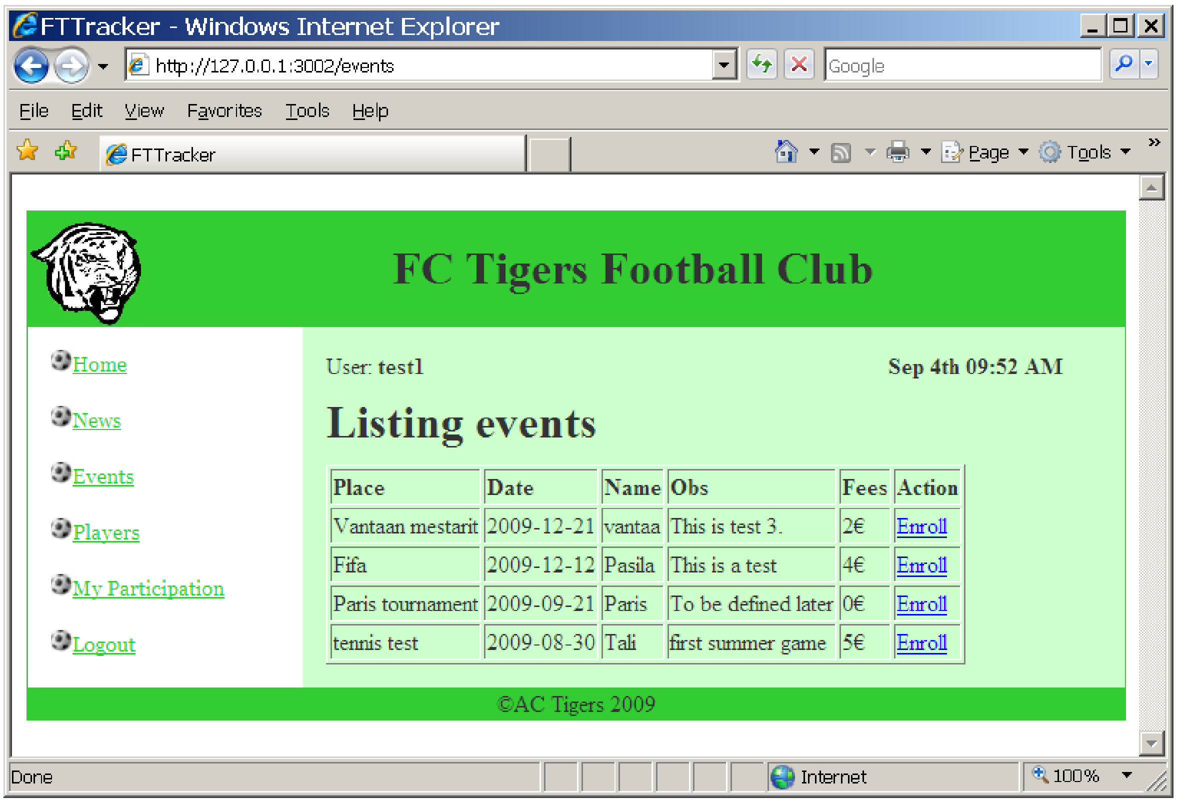
Listing events

Place	Date	Name	Obs	Fees	Action			
Vantaan mestarit	2009-12-21	vantaa	This is test 3.	2€	Show	Edit	Delete	Enroll
Fifa	2009-12-12	Pasila	This is a test	4€	Show	Edit	Delete	Enroll
Paris tournament	2009-09-21	Paris	To be defined later	0€	Show	Edit	Delete	Enroll
tennis test	2009-08-30	Tali	first summer game	5€	Show	Edit	Delete	Enroll

Add Event

©AC Tigers 2009

Admin's view.



FTTracker - Windows Internet Explorer

http://127.0.0.1:3002/events

FC Tigers Football Club

Home
News
Events
Players
My Participation
Logout

User: **test1** **Sep 4th 09:52 AM**

Listing events

Place	Date	Name	Obs	Fees	Action
Vantaan mestarit	2009-12-21	vantaa	This is test 3.	2€	Enroll
Fifa	2009-12-12	Pasila	This is a test	4€	Enroll
Paris tournament	2009-09-21	Paris	To be defined later	0€	Enroll
tennis test	2009-08-30	Tali	first summer game	5€	Enroll

©AC Tigers 2009

Player's view.